CS6360.002 DATABASE DESIGN FINAL PROJECT

Team 4

Database Design for Amazon Company

Members:	XIAORAN GUO	xxg180001
	LI DING	lxd140530
	YING LIU	yxl180063

CS6360.002 Database Design Final Project

DATABASE DESIGN FOR AMAZON COMPANY

I. PROJECT DESCRIPTION

Amazon is the largest e-commerce marketplace and cloud computing platform in the world. It was founded by Jeff Bezos at 1994 and started as an online bookstore but later diversified to sell video, MP3, audiobook downloads/streaming, software, video games, electronics, apparel, furniture, food, toys, and jewelry.

In this project, we design a database system for Amazon.com. Based on the functionalities of Amazon website, this database needs several entities to implement this whole system: Customer, Product, Review, Order, Provider, Ad_place, Shipment_Detail.

II. ENTITIES & REQUIREMENTS

Customer

Customers are people who shop online and place orders, which may contain a list of order_item, say, products. For our design, each customer can place many orders and write reviews about product they bought.

- Attributes: Customer_ID (Key), FirstName, LastName, Email, Address, Login_Name, Login_Password, Mobile_Phone, Prime.

Product

Products are shopping items provided by providers. It can be shipped after order issued. Each product can have many reviews and advertisement on Amazon website.

- Attributes: Product_ID (Key), Product_Name, Price, Description, Stock, Category.

Review

Customer Reviews are meant to give customers genuine product feedback from fellow shoppers. Review is written by customers. Each review is about one product in one specific order.

- Attributes: Review_ID (Key), Star, Comments.

Order

Orders are placed by customers. Each order can have several order items, which reserve the details of product information, and shipment information.

- Attributes: Order_ID (Key), Shipping_Address, Date, Order_Price, Order_Status.

Order_Status is the status about one order, it has 5 values: issued, paid, delivered, completed, cancelled.

Provider

In the US alone, Amazon has over 95 million monthly unique visitors. Providers sell their products to a massive audience of customers by providing products for online shopping.

- Attributes: Provider_ID (Key), Provider_Name, Contact_Name, Phone.

AD_Place

Amazon provides ads display service. To showcase their brand and products on websites, apps, and devices, on and off Amazon, providers can use this service for their product.

- Attributes: AD_ID (Key), Place_Information, Start_Date, End_Date, Style.

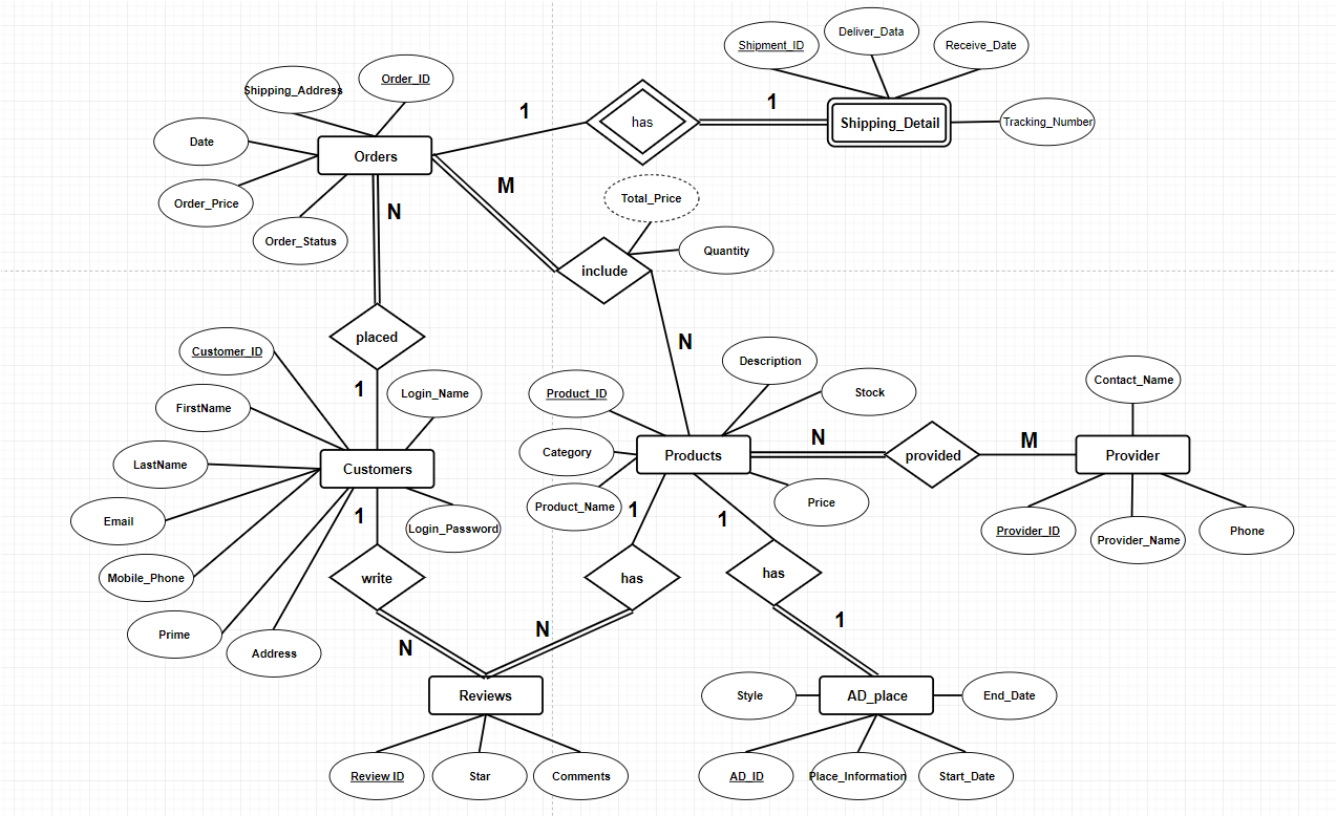
Shipment_Detail

One Shipment_Detail must belong to one order. It keeps the details about the shipment information.

- Attributes: Shipment_ID(Partial Key), Deliver_Date, Receive_Date, Tracking_Number.

III. ER DIGRAM

Based on the relations and entities mentioned in part II, we use draw.io to make our ER diagram.



From the figure above, we can see:

There are 6 entities including Orders, Customers, Reviews, Products, Provider, AD_Place and 1 weak entity which is Shipping_Detail. Total_Price is a derived attribute which could be calculated by Price times Quantity.

By examining the requirements, 7 relationship types are identified. And all are binary relationships

1. PLACED (between CUSTOMER and ORDER) 1 : N
2. WRITE (between CUSTOMER and REVIEW) 1 : N
3. INCLUDE (between ORDER and PRODUCT) M : N
4. HAS_DETAIL (between ORDER and SHIPMENT_DETAIL) 1 : 1
5. PROVIDED (between PRODUCT and PROVIDER) M : N
6. HAS_COMMENT (between PRODUCT and REVIEW) 1 : N
7. HAS_AD (between PRODUCT and AD_PLACE) 1 : 1

IV. MAPPING TO RELATIONAL SCHEMA & NORMALIZATION

When normalizing our relations, we obey the following rules:

a. Make sure each attribute in relations isn't a composite or multivalued attribute. (1NF)

b. Check every non-prime, make sure it is fully functionally dependent on the primary key we defined in each relation. (2NF)

c. Recognize transitive functional dependency and solve it by making a new relation to make sure all attributes depend nothing but the key. (3NF)

1. Customer

Customer_ID	FirstName	LastName	Email	Address	Login_Name	Login_Password	Mobil_phone	Prime

- Primary Key: Customer_ID
- Foreign Keys: None

2. Product

Product_ID	Product_Name	Price	Description	Stock	Category

- Primary Key: Product_ID
- Foreign Keys: None

3. Review

Review_ID	Product_ID	Customer_ID	Star	Comments

- Primary Key: Review_ID
- Foreign Keys:

  FOREIGN KEY (Product_ID) REFERENCES PRODUCTS (Product_ID)

FOREIGN KEY (Customer_ID) REFERENCES CUSTOMERS (Customer_ID)

4. Order

Order_ID	Customer_ID	Shipping_Address	Date	Order_Price	Order_Status

- Primary Key: Order_ID
- Foreign Keys:
  FOREIGN KEY (Customer_ID) REFERENCES CUSTOMERS (Customer_ID)

5. Provider

Provider_ID	Provider_Name	Contact_Name	Phone

- Primary Key: Provider_ID
- Foreign Keys: None

6. AD_Place

AD_ID	Product_ID	Place_Information	Start_Date	End_Date	Style

- Primary Key: AD_ID
- Foreign Keys:
  FOREIGN KEY (Product_ID) REFERENCES PRODUCTS (Product_ID)

7. Shipment_Detail

Order_ID	Shipment_ID	Deliver_Date	Receive_Date	Tracking_Number

- Primary Key: Order_ID, Shipment_ID
- Foreign Keys:
  FOREIGN KEY (Order_ID) REFERENCES ORDERS (Order_ID)

8. Provided

Provider_ID	Product_ID

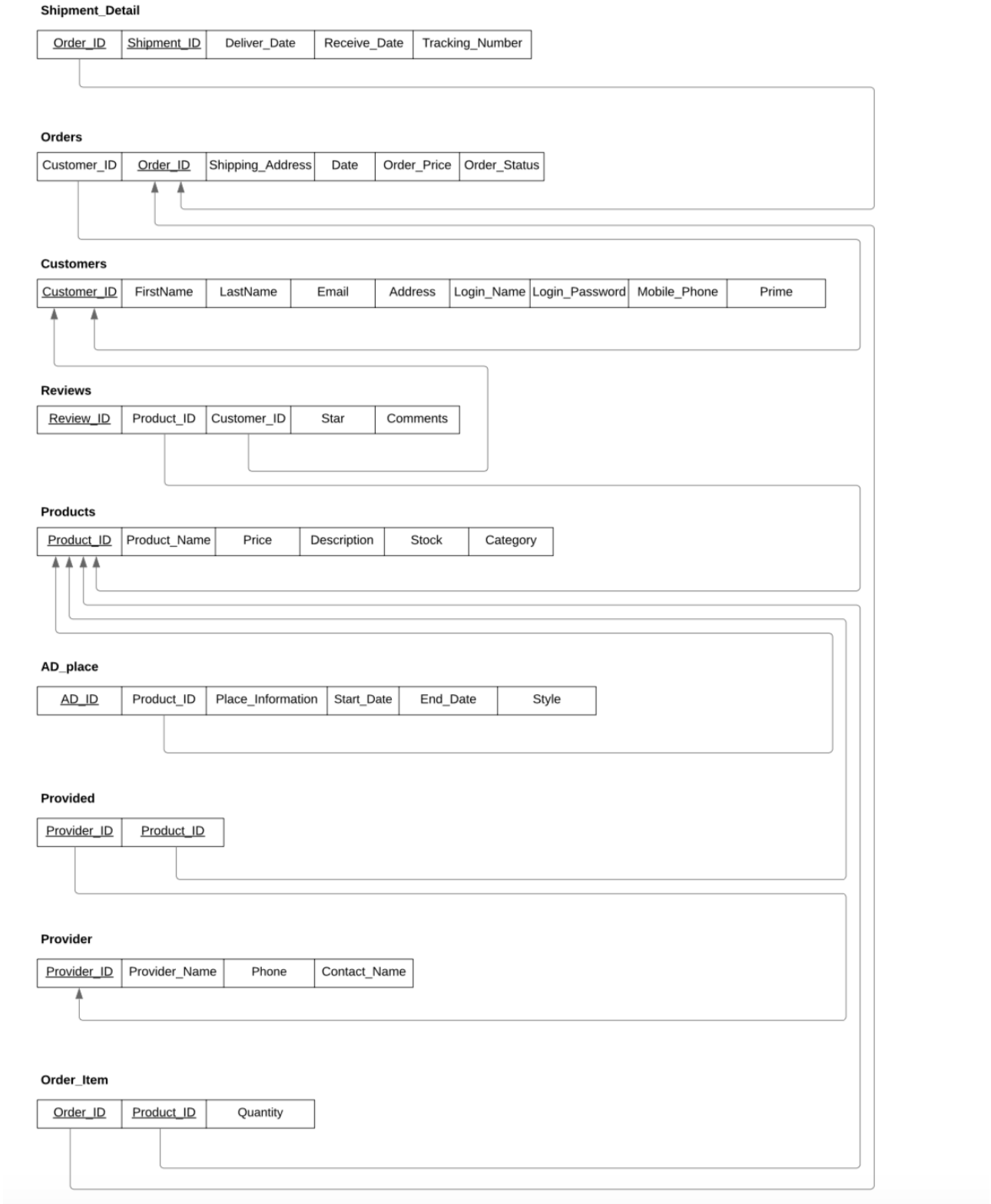
- Primary Key: Provider_ID + Product_ID
- Foreign Keys:
  FOREIGN KEY (Provider_ID) REFERENCES PROVIDER (Provider_ID)
  FOREIGN KEY (Product_ID) REFERENCES PRODUCTS (Product_ID)

9. Order_Item

Order_ID	Product_ID	Quantity

- Primary Key: Order_ID + Product_ID
- Foreign Keys:
  FOREIGN KEY (Order_ID) REFERENCES ORDERS (Order_ID)
  FOREIGN KEY (Product_ID) REFERENCES PRODUCTS (Product_ID)

V. FINAL RELATIONAL SCHEMA



VI. CREATING TABLE (SQL)

Using SQL command to create tables

```
CREATE TABLE SHIPMENT_DETAIL
        (Order_ID              INT              NOT NULL,
         Shipment_ID           INT              NOT NULL,
         Deliver_Date          DATE             NOT NULL,
         Receive_Date          DATE,
         Tracking_Number       INT              NOT NULL,
        CONSTRAINT SHIPPK
         PRIMARY KEY (Order_ID, Shipment_ID),
        CONSTRAINT SHIPODERFK
         FOREIGN KEY (Order_ID) REFERENCES ORDERS (Order_ID)
                        ON DELETE CASCADE);
CREATE TABLE ORDERS
        (Customer_ID           INT              NOT NULL,
         Order_ID              INT              NOT NULL,
         Shipping_Address      VARCHAR (100)    NOT NULL,
         Date                  DATE             NOT NULL,
         Order_Price           FLOAT            NOT NULL,
         Order_Status          VARCHAR (10)     NOT NULL     DEFAULT 'Issued',
        CONSTRAINT ORDERPK
         PRIMARY KEY (Order_ID),
        CONSTRAINT ODERCUSFK
         FOREIGN KEY (Customer_ID) REFERENCES CUSTOMERS (Customer_ID)
                        ON DELETE CASCADE);
CREATE TABLE CUSTOMERS
```

```
(Customer_ID        INT             NOT NULL,
 FirstName          VARCHAR (20)    NOT NULL,
 LastName           VARCHAR (20)    NOT NULL,
 Email              VARCHAR (20)    NOT NULL,
 Address            VARCHAR (100)   NOT NULL,
 Login_Name         VARCHAR (20)    NOT NULL,
 Login_Password     VARCHAR (10)    NOT NULL,
 Mobile_Phone       VARCHAR (10)    NOT NULL,
 Prime              BOOLEAN         NOT NULL    DEFAULT False,
CONSTRAINT CUSPK
 PRIMARY KEY (Customer_ID));
CREATE TABLE REVIEWS
(Review_ID          INT             NOT NULL,
 Product_ID         INT             NOT NULL,
 Customer_ID        INT             NOT NULL,
 Star               INT             NOT NULL,
 Comments           VARCHAR (1000)  NOT NULL,
CONSTRAINT REVIEWPK
 PRIMARY KEY (Review_ID)
CONSTRAINT REVIEWPRODFK
 FOREIGN KEY (Product_ID) REFERENCES PRODUCTS (Product_ID)
                ON DELETE CASCADE
CONSTRAINT REVIEWCUSFK
 FOREIGN KEY (Customer_ID) REFERENCES CUSTOMERS (Customer_ID)
                ON DELETE CASCADE);
CREATE TABLE PRODUCTS
(Product_ID         INT             NOT NULL,
```

```
    Product_Name        VARCHAR (20)        NOT NULL,
    Price               FLOAT               NOT NULL,
    Description         VARCHAR (1000)      NOT NULL,
    Stock               INT                 NOT NULL,
    Category            VARCHAR (10)        NOT NULL,
   CONSTRAINT PRODPK
    PRIMARY KEY (Product_ID));
CREATE TABLE AD_PLACE
   (AD_ID               INT                 NOT NULL,
    Product_ID          INT                 NOT NULL,
    Place_Information   INT                 NOT NULL,
    Start_Date          DATE                NOT NULL,
    End_Date            DATE                NOT NULL,
    Style               VARCHAR (10)        NOT NULL,
   CONSTRAINT ADPK
        PRIMARY KEY (AD_ID)
   CONSTRAINT ADPRODFK
    FOREIGN KEY (Product_ID) REFERENCES PRODUCTS (Product_ID)
            ON DELETE CASCADE);
CREATE TABLE PROVIDED
   (Provider_ID         INT                 NOT NULL,
    Product_ID          INT                 NOT NULL,
   CONSTRAINT PROVEDPK
        PRIMARY KEY (Provider_ID, Product_ID)
   CONSTRAINT PROVEDPROVFK
    FOREIGN KEY (Provider_ID) REFERENCES PROVIDER (Provider_ID)
            ON DELETE CASCADE
```

```
    CONSTRAINT PROVEDPRODFK
     FOREIGN KEY (Product_ID) REFERENCES PRODUCTS (Product_ID)
                ON DELETE CASCADE);
CREATE TABLE PROVIDER
    (Provider_ID          INT            NOT NULL,
     Provider_Name        VARCHAR (50)   NOT NULL,
     Phone                INT            NOT NULL,
     Contact_Name         VARCHAR (20)   NOT NULL,
    CONSTRAINT PROVPK
          PRIMARY KEY (Provider_ID));
CREATE TABLE ORDER_ITEM
    (Order_ID             INT            NOT NULL,
     Product_ID           INT            NOT NULL,
     Quantity             INT            NOT NULL,
    CONSTRAINT ITEMPK
          PRIMARY KEY (Order_ID, Product_ID)
    CONSTRAINT ITEMORDERFK
     FOREIGN KEY (Order_ID) REFERENCES ORDERS (Order_ID)
                ON DELETE CASCADE
    CONSTRAINT ITEMPRODFK
     FOREIGN KEY (Product_ID) REFERENCES PRODUCTS (Product_ID)
                ON DELETE CASCADE);
```

Populating the tables with initial data

```
INSERT INTO CUSTOMERS (Customer_ID, FirstName, LastName, Email, Address, Login_Name,
Login_Password, Mobile_Phone, Prime)
VALUES
('DANIEL', 'KEVIN', 'dk78@outlook.com', '890 High Rd TX', 'daniel007', 'xkll12',
'354245678', False),
```

('HARRY, 'KEVIN', 'dk70@outlook.com', '690 Low Rd TX', 'harry', 'hayxil092', '645678909', False),
('NYU, 'HELLO, 'xmli@gmail.com', '1023 Coit Rd TX', 'nurcan007', 'yurky12', '358845678', True);

```
INSERT INTO PRODUCTS
VALUES
(3421344, 'Baby Shoes', 34.9, 'this is a pair of very comfortable baby shoes.', 20, 'Shoes'),
(5643516, 'Adult Shoes', 44.9, 'this is a pair of very comfortable adult shoes.', 10, 'Shoes');

INSERT INTO PROVIDER (Provider_Name, Contact_Name, Phone)
VALUES
('ALIBABA', 'MAYUN', 126765179),
('TENCENT, 'HUATENG, 267893098),
('CHUIZI, 'LUOYONGHAO, 265372897);

INSERT INTO PROVIDED
VALUES
(1, 3421344),
(1, 5643516);

INSERT INTO AD_PLACE
VALUES
(121, 5643516, 'MAIN PAGE', '20190430', '20190701', 'FANCY');

INSERT INTO ORDERS
VALUES
(2, 199102, '890 High Rd TX', '20190501', 89.8, 'Delivered'),
(3, 199103, '1023 Coit Rd TX', '20190501', 44.9, 'Issued');

INSERT INTO ORDER_ITEM
VALUES
(199102, 5643516, 2),
(199103, 5643516, 1);

INSERT INTO SHIPMENT_DETAIL
VALUES
(199102, 222, '20190501', NULL, 2348918299);

INSERT INTO REVIEWS
VALUES
(100, 5643516, 1, 4, 'It is quite comfortable');
(101, 5643516, 2, 2, 'I don't like it.');
```

VII. PL/SQL

Procedure1: Prime member Black Friday discount

For prime member in 2019 Black Friday, every order price discount is 10% off. Go through ORDERS, if customer is a prime member and Order_Price > or = 25 then the New Order Price = 0.9 * Order_Price.

```
Create or replace PROCEDURE HolidayDiscount AS
        thisOrder Orders%ROWTYPE;
    CURSOR PrimeOrder IS
        SELECT * FROM Orders WHERE Customer_ID IN
(SELECT Customer_ID FROM Customer WHERE Prime == true)
        FOR UPDATE;
BEGIN
    OPEN PrimeOrder;
    LOOP
        FETCH PrimeOrder INTO thisOrder;
        EXIT WHEN (PrimeOrder%NOTFOUND);
        IF (thisOrder.Date = '2019-11-23' && thisOrder.Order_Price > 25) THEN
            UPDATE Orders SET Order_Price = 0.9*Order_Price;
        END LOOP;
CLOSE PrimeOrder;
END;
```

Procedure2: Report Popular Products (based on numbers of reviews)

This procedure goes though products of all the products those have more # of reviews than the average. And print the details of those products.

```
Create or Replace PROCURE ReportPopularProducts AS
    CURSOR DETAIL IS
        SELECT * FROM PRODUCTS WHERE Product_ID IN
(SELECT PRODUCT_ID FROM REVIEWS Where Num_Reviews >avg(Num_reviews) IN
(SELECT Count (Review_ID) AS Num_Reviews, Product_ID FROM PRODUCTS GROUP BY Product_ID) )
        FOR UPDATE;
BEGIN
    OPEN DETAILS;
    LOOP
        FETCH DETAILS INTO thisProduct;
        EXIT WHEN (DETAILS%NOTFOUND);
DBMS_OUTPUT.put_line('Produce ID: '||thisProduct.Product_ID||'; Product_Name:
'||thisProduct.Product_Name||'; Price: '||thisProduct.Price||' ;Description:
'||thisProduct.Description||'; Stock: '||thisProduct.Stock||'; Category: '||thisProduct.Category);
    END LOOP;
    CLOSE DETAILS;
END;
```

Trigger1: Product Stock Update

This trigger will be executed when a new order_item is inserted or updated from Order_Item table. It should update number of stocks in Products table accordingly.

```
CREATE OR REPLACE TIGGER ProductStockUpdate
BEFORE INSERT OR UPDATE ON ORDER_ITEM
FOR EACH ROW

BEGIN
        IF INSERTING THEN
                UPDATE PRODUCTS SET Stock := Stock - :new.Quantity
                WHERE Product_ID = :new.Product_ID;
        END IF;
        IF UPDATING THEN
                UPDATE PRODUCTS SET Stock := Stock - :new.Quantiy + :old.Quantiy
                WHERE Product_ID = :old.Product_ID;
        END IF;
END;
```

Trigger2: Price History

Keep a table to record changes for product price.

```
CREATE TABLE PriceLogs
(
        Log_ID INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
        Product_ID INT NOT NULL,
        NEW_PRICE INT NOT NULL,
        OLD_PRICE INT NOT NULL,
        CHANGE_DATE DATETIME DEFAULT GETDATE() NOT NULL
) ;

CREATE OR REPLACE TRIGGER PriceHistory
AFTER UPDATE ON PRODUCTS
FOR EACH ROW

BEGIN
        IF ( :new.Price != :old.Price ) THEN
                INSERT INTO PriceLogs (Product_ID, NEW_PRICE, OLD_PRICE, CHANGE_DATE)
                VALUES (Product_ID, :new.Price, :old.Price, GETDATE());
        END IF;
END;
```